十、Spring/Spring MVC

90.为什么要使用 spring？

DI 依赖注入 依赖关系更加明显

轻量级，相对于ejb容器来说，ioc容器更趋于轻量级，程序的开发与发布更便捷

测试方便，集合了junit测试单元

面向切面编程，降低业务逻辑和应用服务之间的耦合

集合了众多框架

Web框架 集成了Spring webmvc框架

事务处理 声明式事务和编程性事务 适用大大小小的事务处理场景

异常处理 能够吧特定的异常转化为一致的，unchecked异常

91.解释一下什么是 aop？

面向切面编程。

负责实施切面的框架。它将切面所定义的横切逻辑织入到切面所指定的连接点中

在Spring中，所有方法都可以认为是joinPoint，但是我们并不需要在所有方法上都添加advice。而pointCut就是 提供一组规则来匹配特定的joinPoint，以添加上advice。

实现aop技术：静态代理，动态代理（JDK代理，CGLIB代理）

92.解释一下什么是 ioc？

控制反转。

高层模块不依赖于低层模块实现，低层模块依赖有高层模块所定义的接口

Spring IoC 动态创建、注入依赖对象；管理对象的生命周期；映射依赖关系。

93.spring 有哪些主要模块？

Core：是spring的核心类库。主要实现IoC功能

Web：提供对常见框架Struts1，struts，JSF的支持，Spring能够管理这些框架，将Spring的资源注入给框架，也能在这些框架的前后插入拦截器

MVC：提供了Model-View-Controller的实现

Aop：提供了拦截器机制，并提供了常用的拦截器供用户定义和配置

Context：提供框架式的Bean访问方式，其他程序可以通过Context访问Spring的Bean资源，相当于资源注入。

Dao：Spring 提供对JDBC的支持。

Orm：提供了常用的“对象/关系”映射APIs的集成层。其中包括JPA、JDO、Hibernate 和 iBatis。

94.spring 常用的注入方式有哪些？

构造注入

Setter注入

接口注入

基于注解注入

95.spring 中的 bean 是线程安全的吗？

Spring中的大部分都是无状态的对象，这种不会因为多线程而导致状态被破坏的对象很适合Spring的默认scope，每个单例的无状态对象都是线程安全的。

Spring根本就没有对bean的多线程安全问题做出任何保证与措施。对于每个bean的线程安全问题，根本原因是每个bean自身的设计。不要在bean中声明任何有状态的实例变量或类变量，如果必须如此，那么就使用ThreadLocal把变量变为线程私有的，如果bean的实例变量或类变量需要在多个线程之间共享，那么就只能使用synchronized、lock、CAS等这些实现线程同步的方法了。

96.spring 支持几种 bean 的作用域？

Singleton

Prototype

Request

Session

Global session

97.spring 自动装配 bean 有哪些方式？

@Resource默认是按照名称来装配注入的，只有当找不到与名称匹配的bean才会按照类型来装配注入；

@Autowired默认是按照类型装配注入的，如果想按照名称来转配注入，则需要结合@Qualifier一起使用；

@Resource注解是由J2EE提供，而@Autowired是由Spring提供，故减少系统对spring的依赖建议使用@Resource的方式；

@Resource和@Autowired都可以书写标注在字段或者该字段的setter方法之上

98.spring 事务实现方式有哪些？

声明式事务：建立在AOP之上。本质上是对方法前后进行拦截，在方法开始之前创建或者加入一个事务，在方法执行之后根据执行情况进行提交或回滚事务。一种基于tx和aop名字空间的xml配置文件，另一种是基于@Transactional注解

编程性事务：使用TransactionTemplate或者直接使用低层的PlatformTransactionManager。

并发事务带来的问题：脏读、不可重复读、幻读

不可重复读是由于数据修改引起的，幻读是由数据插入或者删除引起的。

隔离级别：

TransactionDefinition.ISOLATION\_DEFAULT：使用数据库默认的隔离级别

TransactionDefinition.ISOLATION\_READ\_UNCOMMITTED：允许读取尚未提交的数据

TransactionDefinition.ISOLATION\_READ\_COMMITTED（oracle）

TransactionDefinition.ISOLATION\_REPEATABLE\_READ：对同一字段的读取结果一致，除非被本事务更改（Mysql）

TransactionDefinition.ISOLATION\_SERIALIZATION

事务传播行为：

支持当前事务的情况:

TransactionDefinition.PROPAGATION\_REQUIRED：当前存在，加入。不存在，新建

TransactionDefinition.PROPAGATION\_SUPPORTS：当前存在，加入。不存在，非事务

TransactionDefinition.PROPAGATION\_ MANDATORY：当前存在，加入。不存在，抛异常

不支持当前事务的情况：

TransactionDefinition.PROPAGATION\_REQUIRED\_NEW：创建一个新的事务，如果当前存在，则挂起

TransactionDefinition.PROPAGATION\_NOT\_SUPPORTS：以非事务方式运行，如果当前存在，则挂起

TransactionDefinition.PROPAGATION\_NEVER：以非事务方式运行，如果当前存在，则抛异常

TransactionDefinition.PROPAGATION\_NESTED：嵌套事务

100.说一下 spring mvc 运行流程？

客户端提交请求到DispatcherServlet

**DispatcherServlet**收到请求调用HandlerMapping

**HandlerMapping**生成处理器对象和处理器拦截器，返回给DispathcerServlet

DispatcherServlet通过**HandlerAdapter**执行**Handler**（handler，后端控制器）

Controller执行完成返回ModelAndView

HandlerAdapter将ModelAndView返回给DispatcherServlet

DispatcherServlet调用**ViewReslover**，解析ModelAndView并返回View对象

DispatcherServlet对**View**进行渲染（模型数据填充到视图中）

DispatcherServlet响应用户

101.spring mvc 有哪些组件？

如上。

102.@RequestMapping 的作用是什么？

用来处理请求地址映射的注解，可用于类或方法上。用于类上，表示类中的所有响应请求的方法都是以该地址作为父路径。

1、value，method

value：指定请求的实际地址，指定的地址可以是URI Template 模式（后面将会说明）；

method：指定请求的method类型， GET、POST、PUT、DELETE等；

2、consumes，produces

consumes：指定处理请求的提交内容类型（Content-Type），例如application/json, text/html;

produces：指定返回的内容类型，仅当request请求头中的(Accept)类型中包含该指定类型才返回；

3、params，headers

params：指定request中必须包含某些参数值时，才让该方法处理。

headers：指定request中必须包含某些指定的header值，才能让该方法处理请求。

与@RequestParam配合使用：映射请求参数。required表示是否必须,默认为true，defaultValue请求参数的默认值

103.@Autowired 的作用是什么？

默认按类型自动装配注入。为了消除代码中的getter和setter，bean属性中的property

若是一个接口有多个实现，bean里引用的是接口名，则程序会报错。因为spring不知道应该引用哪个实现类，可用@Qualifier指定，注解括号里面的应当是接口实现类的**类名**

@Service：声明Xx.java是一个bean；Xx.java在bean中的id是“xx”，即类名首字母小写

十一、Spring Boot/Spring Cloud

104.什么是 spring boot？

Spring boot是一个轻量级框架，可以完成基于sping的应用程序的大部分配置工作。

105.为什么要用 spring boot？

因为Spring很难配置，我们可以通过使用Spring boot来快速构建Sping应用程序，它只需要极少的配置就。

106.spring boot 核心配置文件是什么？

application：应用级别。用于Spring Boot项目的自动化配置

bootstrap：系统级别。用来加载外部配置，如配置中心的配置信息，也可以用来定义系统不会变化的属性，可以应用到一些加密/解密的场景。它们默认也不能被本地相同配置覆盖。

bootstatp文件的加载先于application文件。

107.spring boot 配置文件有哪几种格式？它们有什么区别？

.properties 和 .yml，它们的区别主要是书写格式不同。

\*.yml加载的属性是有顺序的，但不支持@PropertySource注解来导入配置

yml文件在书写时，需要注意一个地方：冒号与值中间是存在空格的！

108.spring boot 有哪些方式可以实现热部署？

Spring Loaded：引入依赖包，通过启动springloaded启动java文件时，系统会监视classes文件。

几种热部署失效的情况：1、基于第三方框架的注解的修改，如@RequestMapping；2、application.properties的修改；3、log4j的配置文件修改。

Spring-boot-devtools：为应用提供一些开发时特性，包括默认值设置，自动重启，livereload等。

自动重启的原理在于Sping boot使用两个classloader：不该变的类（如第三方jar）由base类加载器加载，正在开发的类由restart类加载器加载。应用重启时，restart类加载器重建。

spring.devtools.restart.exclude 排除

spring.devtools.restart.additional-exclude额外添加

109.jpa 和 hibernate 有什么区别？

Jpa是一种规范，hibernate是它的一种实现。、

Jpa即java持久化接口。与它相关的是ORM技术，对JDBC进行封装。

110.什么是 spring cloud？

基于Spring boot 简化了分布式系统基础设施的开发。

111.spring cloud 断路器的作用是什么？

单个服务出现问题，调用这个服务就会出现线程阻塞，此时若有大量的请求涌入，Servlet容器的线程资源会被消耗完毕，导致服务瘫痪。即服务故障的雪崩效应。

断路器的作用就是在特定服务不可用频率达到一个阈值，断路器就会打开，后续调用会快速返回失败，而不是等待响应超时。

112.spring cloud 的核心组件有哪些？

Eureka：各个服务启动时，Eureka Client都会将服务注册到Eureka Server，并且Eureka Client还可以反过来从Eureka Server拉取注册表，从而知道其他服务在哪里

Ribbon：服务间发起请求的时候，基于Ribbon做负载均衡，从一个服务的多台机器中选择一台

Feign：基于Feign的动态代理机制，根据注解和选择的机器，拼接请求URL地址，发起请求

Hystrix：发起请求是通过Hystrix的线程池来走的，不同的服务走不同的线程池，实现了不同服务调用的隔离，避免了服务雪崩的问题

Zuul：如果前端、移动端要调用后端系统，统一从Zuul网关进入，由Zuul网关转发请求给对应的服务

十二、Hibernate

113.为什么要使用 hibernate？

Hibernate实现了ORM，能够将对象映射成数据库表，从而简化我们的开发！

114.什么是 ORM 框架？

即对象-关系映射，实现JavaBean对象与数据库的映射。

Hibernate是全自动ORM框架，而Mybaits是半自动的。Hibernate可以通过对象关系模型实现对数据库的操作，可自动生成sql。而Mybatis仅有基本字段的映射，对象数据以及对象实际关系仍然需要通过手写sql来实现。

Hibernate数据库一致性大于Mybatis。Hibernate通过它强大的映射结构和hql语言，大大降低了对象与数据库的耦合性，而Mybatis依赖于数据库的sql。

Hibernate的日志系统比Mybatis更完善。

Mybatis更加灵活。

115.hibernate 中如何在控制台查看打印的 sql 语句？

|  |
| --- |
| <hibernate-configuration>  <session-factory>  <property name="**show\_sql**">true</property>  </session-factory>  </hibernate-configuration> |

116.hibernate 有几种查询方式？

117.hibernate 实体类可以被定义为 final 吗？

你可以将Hibernate的实体类定义为final类，但这种做法并不好。因为Hibernate会使用代理模式在延迟关联的情况下提高性能，如果你把实体类定义成final类之后，因为Java不允许对final类进行扩展，所以Hibernate就无法再使用代理了，如此一来就限制了使用可以提升性能的手段。

不过，如果你的持久化类实现了一个接口而且在该接口中声明了所有定义于实体类中的所有public的方法的话，你就能够避免出现前面所说的不利后果。

延迟加载：

当调用Session上的load（）方法加载实体时，就会采用延迟加载

当Session加载某个实体时，会对这个实体中的集合属性值采用延迟加载

当Session加载某个实体时，会对这个实体所单端关联（one-to-one，many-to-one）的另一个实体对象采用延迟加载

Hibernate从数据库获取某一个对象数据、获取某一个对象的集合属性时，获取某一个对象所关联的另一个对象时，由于没有使用该对象的数据，hibernate并不从数据库加载真正的数据。而是为该对象创建一个代理对象，这个代理对象上的所有属性都为默认值，只有在真正的需要该对象的数据时才从数据库中加载数据创建。

实体对象使用延迟加载：

|  |
| --- |
| <hibernate-mapping>  <**class** name=”com.neusoft.entity.User” table=”user” **lazy=”true”**>      ……  </class>  </hibernate-mapping> |

集合类型的延迟加载：

read-only是只读型，缓存不更新，适用于不发生改变的数据，效率最高，事务隔离级别最低，

read-write读写型，缓存在数据变化时触发更新，适用于变化的数据

|  |
| --- |
| <hibernate-mapping>  <class name=”com.neusoft.entity.User” table=”user”>  …  <**set** name=”addresses” table=”address” **lazy=”true”** inverse=”true”>  <key column=”user\_id”/>  <cache usage=”read-only”/>   <!-- <cache usage=”read-write”/>-->  <**one-to-many** class=”com.neusoft.entity.Arrderss”/>  </set>  </class>  </hibernate-mapping> |

属性延迟加载：

|  |
| --- |
| <hibernate-mapping>  <**class** name=”com.neusoft.entity.User” table=”user” >  ……  <property name=”resume” type=”java.sql.Clob” column=”resume” **lazy=”true”/**>  </class>  </hibernate-mapping> |

1. 加载单个实体，如果不需要延迟加载，就可以使用session的get()方法。

2. lazy=false。

用hibernate每次查询完都要关闭session,可关闭session之后，延迟加载的数据就不能查出来了。可在web.xml中配置OpenSessionInView过滤器即可。

|  |
| --- |
| <!-- OpenSessionInView 配置，解决Hibernate延迟加载的问题 -->  <filter>  <filter-name>lazyLoadingFilter</filter-name>  <filter-class>  org.springframework.orm.hibernate3.support.OpenSessionInViewFilter  </filter-class>  </filter>  <filter-mapping>  <filter-name>lazyLoadingFilter</filter-name>  <url-pattern>\*.do</url-pattern>   </filter-mapping> |

118.在 hibernate 中使用 Integer 和 int 做映射有什么区别？

所有的sql使用的默认类型都是null，如果你把POJO中的映射属性类型写为基本类型，当查找不到记录的时候，返回null赋给基本类型就会出错

如果字段是允许为空的，使用包装类。如果不允许为空，使用包装的时候，如果往数据库插入null值，此时就会抛出异常。然后你就可以对异常进行捕获并处理。

使用基本数据类型的时候，如果字段是NULL，那么JDBC会返回0，但是这里会有一个问题。有可能0在你的业务逻辑代表着特定含义，这时候就可能出现一些意想不到的后果。

119.hibernate 是如何工作的？

加载配置文件

创建SessionFactory

打开session

创建事务

事务处理

提交事务

关闭session

关闭SessionFactory

120.get()和 load()的区别？

从返回结果上对比：

load方式检索不到的话会抛出org.hibernate.ObjectNotFoundException异常；get方法检索不到的话会返回null

从检索执行机制上对比：

get方法，hibernate会确认一下该id对应的数据是否存在，首先在session缓存中查找，然后在二级缓存中查找，还没有就查数据库，数据库中没有就返回null。

load方法，首先查找session的persistent Context中是否有缓存，如果有则直接返回 如果没有则判断是否是lazy，如果不是直接访问数据库检索，查到记录返回，查不到抛出异常 如果是lazy则需要建立代理对象，对象的initialized属性为false，target属性为null 在访问获得的代理对象的属性时,检索数据库，如果找到记录则把该记录的对象复制到代理对象的target上，并将initialized=true，如果找不到就抛出异常。

121.说一下 hibernate 的缓存机制？

数据索引 集合类的缓存策略

减少了与数据库的访问，从而提高了数据库访问性能，这就是缓存机制

一级缓存：Hibernate默认的缓存机制，它属于Session级别的缓存机制，也就是说Session关闭，缓存数据消失。

二级缓存：属于SessionFactory级别的缓存，二级缓存是全局性的，应用中的所有Session都共享这个二级缓存。

122.hibernate 对象有哪些状态？

瞬态：刚刚被new创建出来，只是一个普通的类对象

持久态：与一个Hibernate Session关联，在这个状态下，所有属性值的改动，都是可以在事务结束时提交到数据库中

脱管：原本处于持久状态的对象因为其对应的Session被关闭，而失去持久化能力。一旦有Session愿意关联脱管对象，则可以变为持久对象

save()方法：

应该避免在事务之外调用save()方法，否则关联实体(例如employee和address是一对一关系，相互关联)将不会被保存从而导致不一致。很容易忘记在最后调用flush()方法，因为不会有任务的异常或者警告抛出。

hibernate save()方法会立即返回id，原因很可能是调用save()的同时这个实体对象已经被写入数据库(立即执行sql语句insert into)

提交事务或者调用flush()方法，save()方法才会将关联对象也写入数据库。

persist()方法：

persist()方法会将实体对象添加到持久化上下文中，如此被保存的实体后续改变会被记录。如果在提交事务或者会话flush()，对象的属性被重新赋值，那么这个变化也会被保存到数据库中。

persist()方法必须在事务内执行，才能够将数据插入数据库。如果不在事务范围内执行，数据将丢失。

最后，persist()方法返回值是void，也就是说不会返回任何的值。

SaveOrUpdate()方法：

hibernate saveOrUpdate()方法会执行插入或者更新操作。如果该对象在数据库中已经存在则更新，不存在则插入。

saveOrUpdate()方法可以在没有事务的情况下执行，但是如果没有手动调用flush()方法会面临关联对象不被保存的问题

save()方法与saveOrUpdate()方法最大的不同点在于，saveOrUpdate()方法会将实体对象添加到持久化上下文中，该实体的后续改变会被跟踪。

update()方法：

当确定只更新实体信息时使用Hibernate update()方法。update()方法会将实体添加到持久化上下文，实体后续的改变会被跟踪并且当事务提交时这些改变会被保存到数据库中

hibernate update()方法会跟踪实体的改变，在提交事务时保存到数据库中。

merge()方法：

hibernate merge()方法被用于更新数据库中的记录，然而merge()方法通过创建一个传递进来的实体对象副本并且将这个副本作为返回值返回。返回值属于持久化上下文，能够跟踪实体的改变，而传递进来的实体并不能被跟踪。这一点是merge()方法与其他方法最大的不同。

123.在 hibernate 中 getCurrentSession 和 openSession 的区别是什么？

openSession是打开一个新的session，而getCurrentSession则是获取当前线程里的session，如果没有才打开新的。hibernate可以通过session来控制事务，有了getCurrentSession方法意味着可以将对数据库的操作代码放到不同的地方（不同类的方法中），这样事务控制起来极为方便。

124.hibernate 实体类必须要有无参构造函数吗？为什么？

因为生成代理的时候,是CGLIB从你这个实体类继承,然后再依靠反射机制生成实例,必须要提供一个无参的构造方法

十三、Mybatis

125.mybatis 中 #{}和 ${}的区别是什么？

#{} 将传入的数据当成一个字符串

${} 床传入的数据直接拼接到sql中，可以引起sql注入

126.mybatis 有几种分页方式？

使用limit关键字分页

逻辑分页：mybatis自带的分页rowbounds

物理分页：在查询的时候使用limit关键字分页

127.RowBounds 是一次性查询全部结果吗？为什么？

是的

128.mybatis 逻辑分页和物理分页的区别是什么？

逻辑分页在内存中分页

物理分页 通过sql在数据库查询中直接分页查询

129.mybatis 是否支持延迟加载？延迟加载的原理是什么？

MyBatis的延迟加载只是对关联对象的查询有延迟设置，对于主加载对象都是直接执行查询语句的。

直接加载：执行完对主加载对象的select语句，马上执行对关联对象的select查询。

侵入式延迟：执行对主加载对象的查询时，不会执行对关联对象的查询。但当要访问主加载对象的详情时，就会马上执行关联对象的select查询。

深度延迟：执行对主加载对象的查询时，不会执行对关联对象的查询。访问主加载对象的详情时也不会执行关联对象的select查询。只有当真正访问关联对象的详情时，才会执行对关联对象的select查询。

|  |  |  |
| --- | --- | --- |
| 加载策略 | LazyLoadingEnabled | aggressiveLazyLoading |
| 直接加载 | False | False |
| 深度延迟加载 | True | False |
| 侵入式延迟加载 | True | Ture |

130.说一下 mybatis 的一级缓存和二级缓存？

一级缓存是SqlSession级别的缓存。在操作数据库时需要构造sqlSession对象，在对象中有一个数据结构（HashMap）用于存储缓存数据。不同的sqlSession之间的缓存数据区域（HashMap）是互相不影响的。

二级缓存是mapper级别的缓存，多个SqlSession去操作同一个Mapper的sql语句，多个SqlSession可以共用二级缓存，二级缓存是跨SqlSession的。

二级缓存实现：

在全局配置文件中加入：

|  |
| --- |
| <settings>  <setting name="cacheEnabled" value="true"/>  </settings> |

在xxMapper.xml文件中加入（默认PerpetualCache）：

|  |
| --- |
| <cache></cache> |

po类实现Serializable序列化接口。为了将缓存数据取出执行反序列化操作，因为二级缓存数据不一定只存在内存中，有可能存在硬盘中，如果我们要再取这个缓存的话，就需要反序列化了。所以mybatis中的pojo都去实现Serializable接口。

131.mybatis 和 hibernate 的区别有哪些？

132.mybatis 有哪些执行器（Executor）？

133.mybatis 分页插件的实现原理是什么？

使用分页查询的时候，先使用PageHelper.startPage这样的语句在当前线程上下文中设置一个ThreadLocal变量，再利用mybatis提供的拦截器（插件）实现一个com.github.pagehelper.PageInterceptor接口，这个分页拦截器拦截到后会从ThreadLocal中拿到分页的信息，如果有分页信息，这进行分页查询，最后再把ThreadLocal中的东西清除掉。

134.mybatis 如何编写一个自定义插件？